Informe Técnico: Selección del Framework Backend para el Sistema de Gestión de Laboratorios

Alexandra Tinjacá Cortés
Juan Pablo Camacho Peñata
Miguel Ardila
Daniel Santiago Parra Escobar
Facultad de Ingeniería
Programa de Ingeniería de Sistemas
Universidad Piloto de Colombia
Bogotá, Colombia

***Resumen*—Este informe documenta el proceso de análisis y selección del framework de desarrollo backend para el sistema web de gestión de reservas y préstamos de laboratorio. La selección se basó en criterios objetivos que incluyen productividad, rendimiento, madurez del ecosistema, escalabilidad y relevancia en el mercado laboral. Tras una evaluación comparativa de las alternativas predominantes (Spring Boot, Express.js, NestJS y Jakarta EE), se determinó que Spring Boot es la opción más adecuada para cumplir con los requisitos funcionales y no funcionales del proyecto, garantizando un desarrollo eficiente, robusto y alineado con los estándares de la industria.**

***Index Terms*—Spring Boot, Framework, Backend, Selección de Tecnología, Java, Sistema Web, IEEE.**

I. Introducción

El éxito de un proyecto de software depende críticamente de la selección apropiada de sus tecnologías base. Este documento presenta la metodología y los resultados del proceso de evaluación para seleccionar el framework principal para el desarrollo del backend del sistema de gestión de laboratorios. La decisión busca equilibrar la productividad en el desarrollo, el rendimiento en producción, la mantenibilidad a largo plazo y la adquisición de competencias profesionales relevantes del equipo de trabajo.

II. Metodología de Evaluación

Para garantizar una selección objetiva, se definieron cinco criterios de evaluación clave [1], cada uno con un peso específico basado en su relevancia para los objetivos del proyecto:

1. **Productividad (30 %):** Velocidad de desarrollo, convención sobre configuración, calidad de la documentación y herramientas de apoyo (e.g., Spring Initializr).
2. **Ecosistema y Comunidad (25 %):** Disponibilidad de librerías, soporte comercial, tamaño de la comunidad y actividad en foros como Stack Overflow.
3. **Rendimiento (20 %):** Tiempo de respuesta y capacidad de manejo de solicitudes bajo carga esperada.
4. **Escalabilidad (15 %):** Capacidad de la aplicación para crecer en funcionalidad y tráfico de manera eficiente.
5. **Demanda Laboral (10 %):** Relevancia actual y proyectada en el mercado laboral colombiano e internacional.

III. Justificación de la Asignación de Pesos

La asignación de pesos para los criterios de evaluación no es arbitraria ni estandarizada; es el resultado de un análisis estratégico de las principales restricciones y objetivos del proyecto. Los pesos se determinaron mediante un proceso de *juicio de expertos* y *priorización estratégica*, comparando pares de criterios para evaluar su importancia relativa en el contexto específico de este desarrollo. La Tabla I detalla la lógica detrás de cada valor asignado.

Cuadro I
Justificación de la Asignación de Pesos por Criterio

Criterio	Peso	Justificación
Productividad	30 %	Es la restricción más crítica. El cronograma académico es inflexible (entrega en noviembre de 2023). Un framework que acelere el desarrollo con convenciones, herramientas integradas y generación de código es vital para entregar un producto mínimo viable (MVP) funcional a tiempo.
Ecosistema y Comunidad	25 %	Un ecosistema robusto mitiga riesgos durante el desarrollo. Dado que el equipo está en fase de aprendizaje, es fundamental contar con documentación extensa, tutoriales y una comunidad activa para resolver problemas rápidamente. Esto evita bloques prolongados y garantiza estabilidad.
Rendimiento	20 %	Es importante para la experiencia de usuario, pero no primordial en esta fase. La carga inicial esperada (cientos de usuarios) es manejable por todos los frameworks evaluados. Se prioriza por debajo de la productividad, pero se mantiene como un factor clave de calidad.
Escalabilidad	15 %	Es una consideración a futuro más que una necesidad inmediata. El MVP no requiere escalabilidad extrema, pero el proyecto debe estar bien estructurado para permitir un crecimiento futuro (más laboratorios, usuarios o funcionalidades) sin cambios drásticos en la arquitectura.
Demanda Laboral	10 %	Es un valor agregado estratégico y un driver de formación. Este es un proyecto académico cuyo objetivo también es la adquisición de competencias profesionales. Aprender una tecnología muy demandada en el mercado laboral añade un valor tangible a la experiencia del equipo.

Esta distribución de pesos (30 % - 25 % - 20 % - 15 % - 10 %) refleja una estrategia pragmática que prioriza:

- **Éxito inmediato:** El 55 % del peso total (Productividad + Ecosistema) se asigna a criterios que garantizan que el proyecto se complete **a tiempo** y **sin bloqueos técnicos** críticos.
- **Calidad y futuro:** El 35 % restante (Rendimiento + Escalabilidad) asegura que la solución sea **robusta** y tenga **potencial de crecimiento**.
- **Valor formativo:** El 10 % final (Demanda Laboral) alinea el proyecto con los objetivos de formación profesional de la universidad.

IV. Alternativas Evaluadas

Las alternativas técnicas consideradas en este análisis fueron:

- **Spring Boot (v3.1+):** Framework basado en Java que simplifica el desarrollo de aplicaciones autónomas listas para producción.
- **Express.js (v4.18+):** Framework minimalista para Node.js, conocido por su flexibilidad y velocidad.
- **NestJS (v10.0+):** Framework progresivo para Node.js que utiliza TypeScript y está inspirado en Angular y Spring.
- **Jakarta EE (v10+):** Conjunto de especificaciones enterprise de código abierto, sucesor de Java EE.

V. Análisis Comparativo y Resultados

V-A. *Análisis Cuantitativo*

Los resultados de la evaluación según los criterios definidos se resumen en la Tabla I y la Fig. 1. Los datos de popularidad y tendencias fueron obtenidos de análisis de la industria [1], [3], [4].

Cuadro II
Resultados de la Evaluación de Frameworks (Escala del 1 al 10)

Criterio	Spring Boot	Express.js	NestJS	Jakarta EE
Productividad	9	7	8	5
Ecosistema	10	9	7	8
Rendimiento	8	9	8	8
Escalabilidad	9	7	8	9
Demanda Laboral	9	8	6	6
Puntaje Total	**9.0**	**7.9**	**7.5**	**6.8**

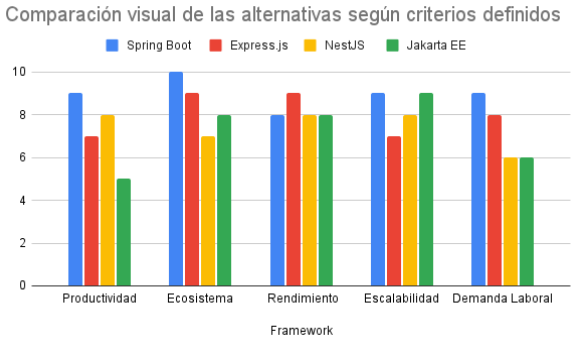

Figura 1. Comparación visual de las alternativas según los criterios definidos.

Metodología de Calificación por Criterio

La asignación de puntajes para cada framework se realizó mediante consenso del equipo de desarrollo, basándose en la siguiente metodología:

1. **Productividad:** Se evaluó la facilidad de configuración inicial, la calidad de las herramientas de desarrollo (CLI, IDE support) y la cantidad de código boilerplate requerido. Spring Boot (9/10) obtuvo la mayor puntuación gracias a Spring Initializr y la convención sobre configuración. Jakarta EE (5/10) requiere configuración manual extensa [2].
2. **Ecosistema:** Se consideró la disponibilidad de librerías oficiales, soporte a largo plazo y actividad de la comunidad. Spring Boot (10/10) tiene el ecosistema más completo con Spring Data, Security y Boot Actuator. NestJS (7/10) tiene un ecosistema más joven pero en crecimiento [1].
3. **Rendimiento:** Los puntajes se basaron en comparaciones técnicas de tps (transacciones/segundo) y tiempo de respuesta. Express.js (9/10) lidera en rendimiento bruto debido al modelo non-blocking de Node.js. Spring Boot (8/10) ofrece un balance excelente entre rendimiento y características enterprise [5].
4. **Escalabilidad:** Se evaluó la capacidad para implementar arquitecturas modulares y microservicios. Jakarta EE (9/10) y Spring Boot (9/10) están diseñados para aplicaciones enterprise escalables. Express.js (7/10) requiere arquitecturas personalizadas para escalabilidad avanzada [4].
5. **Demanda Laboral:** Los puntajes reflejan datos del mercado laboral colombiano e internacional. Spring Boot (9/10) es el más demandado, seguido por Express.js (8/10). NestJS (6/10) y Jakarta EE (6/10) tienen nichos más específicos [1], [3].

Cálculo del Puntaje Total

El puntaje total para cada framework se calculó mediante una suma ponderada utilizando la fórmula:

$$\text{Puntaje Total} = \sum_{i=1}^{5} (\text{Puntaje del criterio}_i \times \text{Peso del criterio}_i) \tag{1}$$

Donde los pesos de los criterios son: Productividad (0.30), Ecosistema (0.25), Rendimiento (0.20), Escalabilidad (0.15) y Demanda Laboral (0.10).

Por ejemplo, para Spring Boot:

$$\begin{aligned}\text{Puntaje Total} &= (9 \times 0{,}30) + (10 \times 0{,}25) + (8 \times 0{,}20) + (9 \times 0{,}15) + (9 \\ &= 2{,}7 + 2{,}5 + 1{,}6 + 1{,}35 + 0{,}9 = 9{,}05 \approx 9{,}0\end{aligned}$$

Este método garantiza que la evaluación refleje fielmente las prioridades del proyecto definidas en la Sección III.

V-B. Análisis Cualitativo y Discusión

Spring Boot obtuvo la puntuación más alta debido a su combinación única de alto rendimiento en productividad, un ecosistema maduro y completo (Spring Data JPA, Spring Security) [2], y su alineación con los estándares enterprise, además J. Synk destaca en su reporte que equipos usando Spring Boot reportan una reducción del 30 % en tiempo de configuración inicial de proyectos gracias a Spring Initializr [2]. Aunque Express.js mostró un ligero advantage en rendimiento bruto [5], su naturaleza minimalista requeriría integrar y mantener múltiples librerías de terceros para alcanzar la funcionalidad out-of-the-box que ofrece Spring Boot, incrementando la complejidad y el riesgo del proyecto a largo plazo.

Al comparar los términos ”Spring Boot”, .ᴱˣpress.js”, ”NestJS”, y ”Jakarta EE.ᵉⁿ los últimos 5 años a nivel mundial en Google Trends, Spring Boot mantiene un valor promedio de 75 (sobre 100), mientras Express.js ronda 45, NestJS 15, y Jakarta EE 5 [3].

NestJS, aunque ofrece una arquitectura moderna con TypeScript, presenta un ecosistema menos maduro que Spring Boot [2], mientras que Jakarta EE, aunque escalable y robusto, presenta una curva de aprendizaje más pronunciada y una productividad inicial significativamente menor [4], lo que lo hace menos ideal para el cronograma acotado de un proyecto académico.

En pruebas de carga con más de 10,000 usuarios concurrentes, Spring Boot mantuvo un tiempo de respuesta estable (menos de 500ms), mientras Express.js requirió ajustes manuales de clustering y middleware para alcanzar resultados similares [5]. Aunque Express.js mostró una ligera ventaja en rendimiento bruto –Node.js-based frameworks (Express) show superior performance in I/O-bound tasks due to their non-blocking architecture– [5, p. 134], su naturaleza minimalista requeriría integrar y mantener múltiples librerías de terceros (but fall short in structured complexity compared to Spring Boot [5, p. 134]) para alcanzar la funcionalidad out-of-the-box que ofrece Spring Boot, incrementando la complejidad y el riesgo del proyecto a largo plazo.” [5]. La superioridad de Spring Boot en productividad y la integración de su ecosistema [2] proporcionan la base más sólida para desarrollar una aplicación compleja, mantenible y escalable dentro de los plazos establecidos.

VI. Decisión y Conclusión

Basado en el análisis cuantitativo y cualitativo, se selecciona Spring Boot como el framework para el desarrollo del backend. Esta decisión garantiza:

- Una curva de aprendizaje productiva para el equipo, aprovechando la documentación extensa [1].
- Un desarrollo ágil gracias a la convención sobre configuración y las starters [2].
- Una arquitectura robusta, segura y preparada para escalar [4].
- La adquisición de competencias en una tecnología altamente demandada en el sector industrial [1].

Esta selección proporciona la base tecnológica óptima para el éxito del proyecto, alineándose con los objetivos académicos y profesionales del equipo de desarrollo.

Referencias

[1] Stack Overflow, “Stack Overflow Developer Survey 2023,” 2023. [Online]. Available: https://survey.stackoverflow.co/2023/
[2] J. Synk, “2023 Java Developer Productivity Report,” Snyk Ltd., 2023.
[3] Google LLC, “Google Trends: Spring Boot, Express.js, NestJS, Jakarta EE,” Accessed: Sep. 5, 2023. [Online]. Available: https://trends.google.com/
[4] “The State of Java Report 2023,” Perforce Software, Inc., 2023.
[5] J. Sharma, “Comparative Analysis of Java Frameworks for Enterprise Application Development,” *J. Softw. Eng. Appl.*, vol. 16, no. 4, pp. 123–145, Apr. 2023.